COMPANY CONFIDENTIAL

HEWLETT-PACKARD

GENERAL PURPOSE HP-IL INTERFACE

INTEGRATED CIRCUIT

1LB3-0003

1981 MAY 15

FIGURE 2.1: EXAMPLE HP-IL IMPLEMENTATION

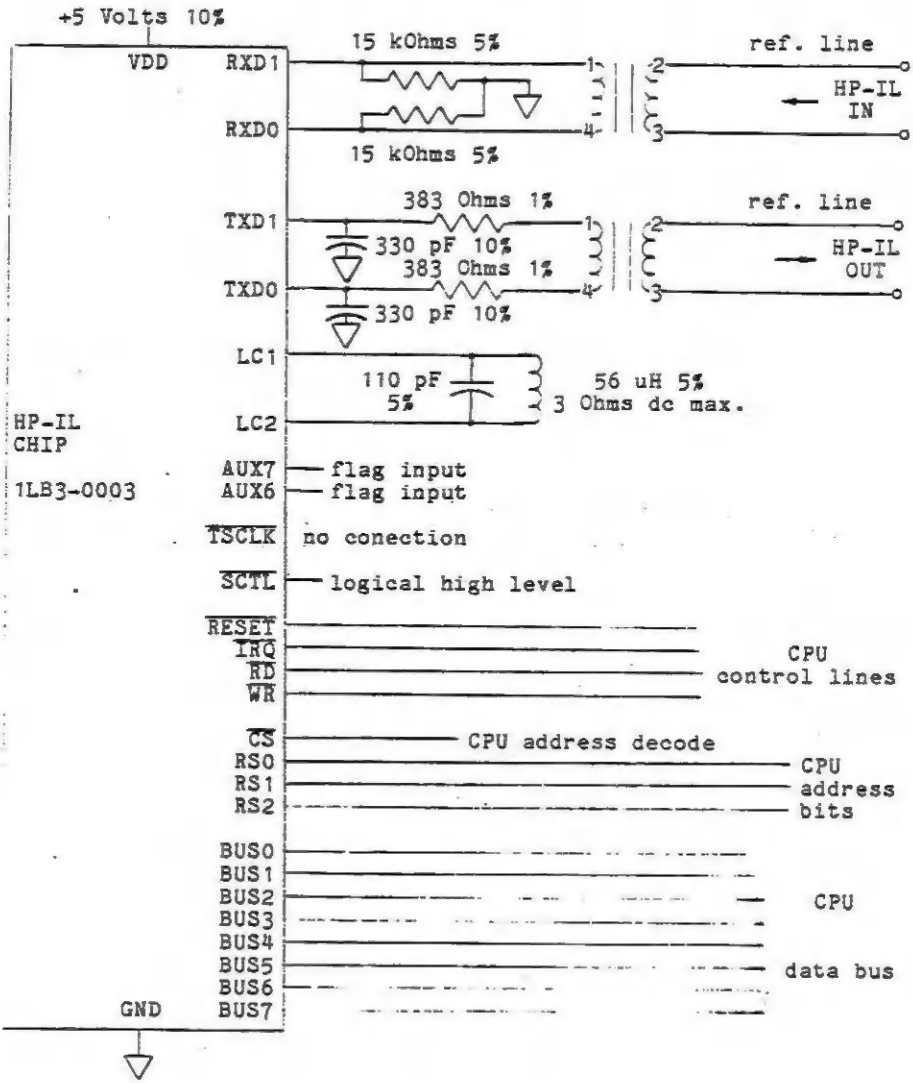


TABLE OF CONTENTS

FIGURES

TABLES

1. GENERAL DESCRIPTION AND FEATURES

This document describes the HP-IL interface chip, hereafter referred to as the chip. This manual is intended primarily for the designer of a device which is to communicate via HP-IL, but may be useful to the sophisticated system user as well. It presumes a complete understanding of the document entitled "HEWLETT-PACKARD INTERFACE LOOP" which defines the functional, electrical, and mechanical aspects of the interface system.

This chip serves the purpose of converting signals back and forth from HP-IL to a microprocessor compatible data bus with appropriate control lines. An external microprocessor, hereafter referred to as the CPU, and the chip together form a complete interface between the device and HP-IL. The microprocessor usually serves the dual function of controlling both the chip and the device. The following list highlights the significant features of the chip:

* CMOS technology, 4.4 to 5.5 Volt single supply, standard 28 or 40 pin DIP package.
* Chip permits implementing Controller, Talker, or Listener functions or any combination except Talker-Listener simultaneously.
* The data bus lines and register address lines are TTL compatible (however, the read, write, and chip select control lines may require external pull-ups for a complete TTL interface).
* Frames which the chip sources are automatically error-checked when they return and the CPU is notified if an error is detected.
* Both the normal handshake and the CMD-RFC handshake are handled automatically by the chip.
* In most circumstances, frames which do not affect this device are automatically retransmitted without the need for CPU intervention.
* All interrupts are fully programmable.
* The chip may be set to send service request or respond to parallel poll automatically.
* Two 8 bit general purpose scratchpad registers are included on the chip.
* Eight flag inputs (two in the 28 pin version) can be read by the CPU.

2. ELECTRICAL DESCRIPTION

The first section of this chapter presents a typical HP-IL implementation to give the designer an approximate idea of the electrical hardware support necessary for the chip. The following section gives the pin-outs for both versions of the chip together with the basic function for each pin. Detailed electrical specifications are contained in the final section.

2.1 Hardware Support

Figure 2.1 is a schematic diagram showing typical interface circuitry. The CPU is not shown here as any of a number of commonly available microprocessors would be suitable for use with the chip.

The signals from the previous device come into the receiver inputs (RXD0, RXD1) through a small pulse transformer which provides a voltage step-up and isolates this device from the loop. The resistors to ground from the receiver inputs provide the proper load for the loop. Line length and parasitic capacitance must be kept to an absolute minimum on these pins.

Each of the transmitter outputs (TXD0, TXD1) passes through a simple low-pass filter and impedance matching network consisting of a capacitor to ground and a series resistor. The signal then goes to another pulse transformer which steps down the voltage to the proper loop level and isolates the device.

The chip oscillator frequency is controlled with an external parallel LC network connected to LC1 and LC2. These lines should be kept as short as possible to minimize parasitic effects. It is the designer's responsibility to see that the oscillator runs at the correct frequency. The 110 pF capacitor value may need to be adjusted up or down to compensate for parasitics.

The CPU data bus connects to BUS0-BUS7. Normally, the three low order address bits of the microprocessor will be connected to RS0-RS2 with some external address decode logic feeding the chip select pin ($\overline{CS}$). The read, write, reset, and interrupt pins ($\overline{RD}$, $\overline{WR}$, $\overline{RESET}$, $\overline{IRQ}$) are usually tied to their corresponding control lines from the CPU. The data bus lines and the register select lines are TTL compatible. For the other control lines, however, pull-up resistors will be necessary if TTL is used to drive them in order to guarantee good noise immunity for the high level.

For the 28 pin version shown here there are four miscellaneous lines. AUX6 and AUX7 are simple flag inputs which the CPU can read. These may be left open if they are not used. Since the chip oscillator is used in this example, no external

clock is needed and so $\overline{\text{TSCLK}}$ is left open also. Unless the device is to be the HP-IL system controller, the $\overline{\text{SCTL}}$ pin should be connected to a high level.

The chip interrupt output is an n-channel open-drain device. The pull-up resistor necessary on this line is not shown.

It is important to remember that the read, write, chip select, reset, interrupt, and external clock lines are active low. All others are active high.

The 28 pin version is used in the example as it is much more economical than the 40 pin version. The 40 pin package should only be used if the additional chip control lines brought out in that version are necessary to the design.

2.2 Chip Pin Description

Table 2.1 contains the pin connection lists for the 40 pin version and the 28 pin version. Table 2.2 gives a brief description of each line and its function.

TABLE 2.1: CHIP PIN-OUTS

40 PIN VERSION

1.	VDD	40.	$\overline{\text{IRQ}}$
2.	$\overline{\text{ILS}}$	39.	$\overline{\text{CS}}$
3.	$\overline{\text{INTL}}$	38.	$\overline{\text{WR}}$
4.	RS2	37.	$\overline{\text{RD}}$
5.	RS1	36.	$\overline{\text{RESET}}$
6.	RS0	35.	$\overline{\text{RTRN}}$
7.	BUS7	34.	LC2
8.	BUS6	33.	LC1
9.	BUS5	32.	$\overline{\text{TSCLK}}$
10.	BUS4	31.	$\overline{\text{SCTL}}$
11.	GND	30.	TXD1
12.	AUX0	29.	
13.	AUX1	28.	
14.	AUX2	27.	TXD0
15.	AUX3	26.	
16.	BUS3	25.	RXD0
17.	BUS2	24.	RXD1
18.	BUS1	23.	AUX7
19.	BUS0	22.	AUX6
20.	AUX4	21.	AUX5

28 PIN VERSION

1.	VDD	28.	$\overline{\text{CS}}$
2.	$\overline{\text{IRQ}}$	27.	$\overline{\text{WR}}$
3.	RS2	26.	$\overline{\text{RD}}$
4.	RS1	25.	$\overline{\text{RESET}}$
5.	RS0	24.	LC2
6.	BUS7	23.	LC1
7.	BUS6	22.	$\overline{\text{TSCLK}}$
8.	BUS5	21.	$\overline{\text{SCTL}}$
9.	BUS4	20.	TXD1
10.	GND	19.	TXD0
11.	BUS3	18.	RXD0
12.	BUS2	17.	RXD1
13.	BUS1	16.	AUX7
14.	BUS0	15.	AUX6

TABLE 2.2: PIN DESCRIPTION

pin	description
VDD	Positive supply voltage, 5 Volts nominal.
GND	Ground. All voltage measurements are referenced to this pin.
RXD0-RXD1	HP-IL inputs. Because the input voltage may swing above VDD, these pins have the special protection circuit. These are Schmitt trigger inputs.
TXD0-TXD1	HP-IL outputs. The idle state is both outputs low (these outputs are not tri-state).
RS0-RS2	Register select (address) inputs, TTL compatible. For historical reasons, RS0 and RS1 have the special protection circuit while RS2 has the normal protection circuit.
BUS0-BUS7	Bidirectional data bus lines, TTL compatible. The output circuit provides intrinsic protection for the input (even when the output is off) so no protection circuit is necessary. The outputs are turned on when both the read and chip select lines are low.
$\overline{CS}$	Chip select input, normal protection circuit. This signal is active low.
$\overline{RD}$	Read input, normal protection circuit, active low.
$\overline{WR}$	Write input, normal protection circuit, active low.
$\overline{RESET}$	Reset input, normal protection circuit, active low.
$\overline{IRQ}$	Interrupt request output. This output is an N-channel open drain device. An external pull-up is necessary for the high level. Even though this is an output, it has the normal protection circuit. This line is also active low.
LC1-LC2	Oscillator lines, normal protection circuit. If the LC network is connected here for the chip oscillator, the $\overline{TSCLK}$ input should be left open.
$\overline{TSCLK}$	External oscillator input, normal protection circuit. If this line is used, LC2 should be tied high and LC1 should be left open. This input has a small pull-up device to drive it high if it is not connected. The external clock signal could be gated with $\overline{RESET}$, if desired.

TABLE 2.2: PIN DESCRIPTION (continued)

pin	description
AUX0-AUX7	Auxiliary inputs, normal protection circuit. These are the flag lines that can be read by the CPU. A small pull-up device is included on each line to drive it high if it is not connected externally.
$\overline{\text{SCTL}}$	System controller input, normal protection circuit, active low. If this input is tied high externally, the chip will power on assuming that it is not the HP-IL system controller, and vice versa.
$\overline{\text{RTRN}}$	Retransmit only input, normal protection circuit, active low. If this line is driven low, the chip will be a simple repeater. A small pull-up is included to drive the input high if it is not connected externally. This line is only available on the 40 pin version.
$\overline{\text{INTL}}$	Interrupt latch output, active low. This is a simple latch output which can be set low by any interrupt or by a low on $\overline{\text{ILS}}$ or $\overline{\text{RESET}}$. The latch is cleared by any write to register 7. This line can be used to control power-up circuitry for the CPU. This is an N-channel open drain device and an external pull-up is necessary for the high level. Even though this is an output, the normal protection circuit is included. This line is only available on the 40 pin version.
$\overline{\text{ILS}}$	Interrupt latch set input, normal protection circuit, active low. A small pull-up device pulls this line high if it is not connected externally. This pin is only available in the 40 pin version.

NOTE: CMOS inputs are very susceptible to damage from electrostatic discharge (ESD). For this reason, a protection circuit is included on each input to reduce the possibility of damage. The normal protection circuit consists of reverse biased diodes to both VDD and GND. If the voltage rises more than a diode drop above the supply or falls more than a diode drop below ground, it is prevented from going farther and destroying the input. In the case of the HP-IL inputs, however, the normal input voltage may rise above VDD, so a special protection circuit is used which has only the diode to ground. Output drivers include intrinsic diodes and do not normally require any addtional protection. The open drain outputs do include the normal protection circuit, however.

2.3 Electrical Specifications

All voltages throughout this specification are measured with respect to the GND pin of the chip.

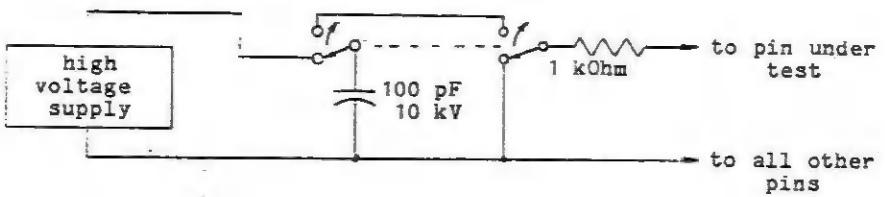
ABSOLUTE MAXIMUM RATINGS

Supply voltage VDD	-0.5 V to 10.0 V
Voltage on any pin (note 2)	-0.5 V to VDD+0.5 V
Storage temperature	-50 C to 150 C
ESD voltage on any pin (note 3)	-1000 V to 1000 V

1. Compliance with these values only guarantees no permanent damage, not necessarily correct device operation.

2. The voltage on RXD0 and RXD1 is allowed to swing above VDD to a maximum of approximately 15 V without damage.

3. ESD test circuit:



RECOMMENDED OPERATING CONDITIONS

Supply voltage VDD	4.4 V to 5.5 V
Ambient temperature	0 C to 65 C
Relative humidity	0% to 90%
Oscillator period (note 4)	475 nS to 550 nS
VDD standby current (note 5)	1 uA maximum
VDD idle current (note 6)	2.5 mA maximum
VDD transmit current (note 7)	approx. 5 mA maximum

4. When the oscillator period is measured at LC1 or LC2, a low capacitance probe (1 pF maximum) must be used. If the chip oscillator is not used, the external clock signal must have rise and fall times less than 50 nS.

5. Standby current is measured at maximum VDD with $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ tied to VDD; RS0-RS2, RXD0-RXD1, BUS0-BUS7, $\overline{RESET}$, and $\overline{SCTL}$ are tied to GND; all other pins are left open.

6. Idle current is measured at maximum VDD with the chip oscillator at maximum frequency; $\overline{RESET}$, $\overline{CS}$, $\overline{RD}$, and $\overline{WR}$ are tied to VDD; RS0-RS2, RXD0-RXD1, BUS0-BUS7, and $\overline{SCTL}$ are tied to GND; all other pins are left open.

7. Transmit current is measured at maximum VDD with the chip oscillator at maximum frequency; $\overline{\text{RESET}}$, $\overline{\text{CS}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ are tied to VDD; RS0-RS2, BUS0-BUS7, and $\overline{\text{SCTL}}$ are tied to GND; RXD0-RXD1 are tied to TXD0-TXD1 respectively, with a 1.6 kOhm load resistor connected across TXD0 and TXD1; all other pins are open; HP-IL frames are being retransmitted at the maximum rate.

STATIC INPUT PARAMETERS

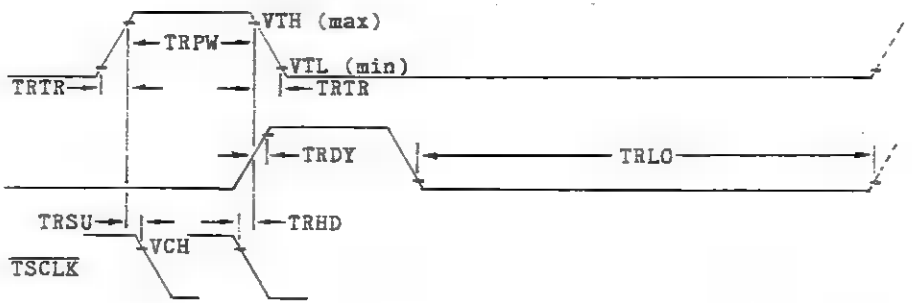
VIL	low logic level input voltage for RS0-RS2, BUS0-BUS7	0.8 V maximum
VIH	high logic level input voltage for RS0-RS2, BUS0-BUS7	2.4 V minimum
VTL	low logic level threshold voltage for RXD0-RXD1	1.2 V to 3.3 V
VTH	high logic level threshold voltage for RXD0-RXD1	2.2 V to 4.2 V
VHS	threshold hysteresis voltage (VTH-VTL) for RXD0-RXD1	0.5 V minimum
VCL	low logic level input voltage for all other inputs	0.2 VDD maximum
VCH	high logic level input voltage for all other inputs	0.8 VDD minimum
ILB	leakage current for BUS0-BUS7 (outputs off, pin at VDD or GND)	1 uA maximum
ILI	leakage current for other inputs (pin at VDD or GND except inputs with pull-ups not measured at GND)	0.1 uA maximum
IPU	pull-up current (pin at GND)	10 uA to 200 uA
CIN	input capacitance	8 pF maximum

STATIC OUTPUT PARAMETERS

VOL	low logic level output voltage for BUS0-BUS7, $\overline{\text{IRQ}}$, $\overline{\text{INTL}}$ at sink current of 2.0 mA (1 TTL load or 5 LSTTL loads)	0.4 V maximum
VOH	high logic level output voltage for BUS0-BUS7 at source current of 300 uA	2.7 V minimum
ITL	TXD0-TXD1 sink current at 0.5 V	5.5 mA to 19.2 mA
ITH	TXD0-TXD1 source current at VDD-0.5 V	5.5 mA to 19.2 mA

HP-IL INPUT TIMING PARAMETERS

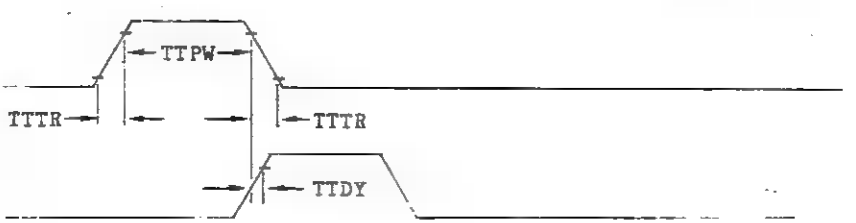
The traces in the timing diagram represent the signal driven on the two RXD0-RXD1 inputs from the interface loop. The setup and hold time from the external clock are specified for testing purposes only. In normal operation the input pulses on RXD0 and RXD1 are totally asynchronous to the clock.



TRPW	Input pulse width	650 nS to 1500 nS
TRTR	Input pulse transition time	300 nS maximum
TRDY	Input pulse delay time	0 nS to 300 nS
TRLO	Input pulse low time	1300 nS minimum
TRSU	Setup time	50 nS minimum
TRHD	Hold time	50 nS minimum

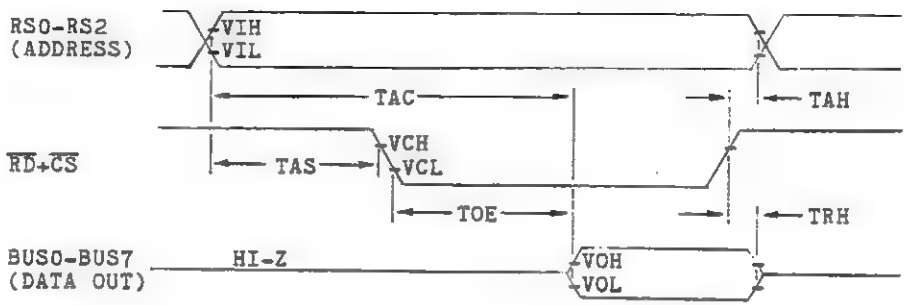
HP-IL OUTPUT TIMING PARAMETERS

The traces in this timing diagram represent the pulses from the two TXD0-TXD1 outputs to the interface loop. The values are measured from the 0.5 V and VDD-0.5 V levels on the waveform with ■ 330 pF load on each of the two outputs.



TTPW	Output pulse width	950 nS to 1200 nS
TTTR	Output pulse transition time	120 nS maximum
TTDY	Output pulse delay time	0 nS to 120 nS

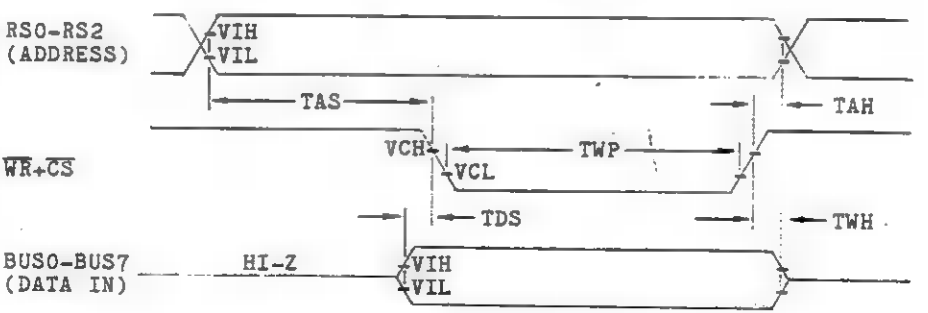
READ CYCLE TIMING PARAMETERS



TAS	Address setup time	50 nS minimum
TAC	Address access time	350 nS maximum
TOE	Output enable time	200 nS maximum
TAH	Address hold time	20 nS minimum
TRH	Read data hold time	50 nS minimum

These values are measured with ■ 75 pF load on each of BUS0-BUS7. The data on BUS0-BUS7 is valid at the end of TAC or TOE, whichever ends later in time. Address setup and hold times must be specified for the read cycle as well as the write cycle to prevent inadvertant reads of R2 (which clears FRAV or FRNS and loads the control bits from R1R to R1W).

WRITE CYCLE TIMING PARAMETERS



TAS	Address setup time	50 nS minimum
TDS	Write data setup time	0 nS minimum
TWP	Write pulse width	300 nS minimum
TAH	Address hold time	20 nS minimum
TWH	Write data hold time	20 nS minimum

The 1LB3-0003 is a standard 28 pin plastic dual in-line chip package. The 28 pin version is strongly recommended wherever possible as the 40 pin version is nearly double the cost. The 28 pin version is missing the $\overline{\text{RTRN}}$, $\overline{\text{ILS}}$, $\overline{\text{INTL}}$, and AUX0-AUX5 pins. Few applications will have a need for these.

3. FUNCTIONAL DESCRIPTION

As mentioned earlier, all communication between the CPU and the chip (and hence the interface loop) takes the form of either memory or I/O reads and writes from the CPU on its eight bit data bus. These data transfers are controlled by the CPU control lines: $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, $\overline{IRQ}$, and $\overline{RESET}$ or their equivalents. The chip has eight memory locations (or I/O registers) which not only send and receive data, but may initiate other actions when the read or write operation is performed, such as transmitting a frame on the interface loop or setting a service request bit in the next frame which comes in.

The first section of this chapter provides the programmer's model of the registers with their general definitions and function. Subsequent sections deal with the bits and registers in detail.

3.1 Register Definition

The eight chip registers may be referred to by name or by address, for example, register 4 and loop address register both indicate the register addressed when RS2, RS1, and RS0 are 1, 0, and 0 respectively. It is important to note that several of the registers access different bits on read than they do for a write. That is, some bits are either read-only or write-only while others are normal read-write bits. Occasionally a short form notation will be used to refer to a particular set of bits. R5RW indicates both the read and write portions of register 5, one of the scratchpad registers. R1R refers only to the read bits of register 1, the interrupt register. R1W would indicate only the write bits of that same register. Figure 3.1 and table 3.1 give the names and mnemonics of the registers and bits.

Register 0 is called the status register. The CPU sets bits in this register to put the chip in controller mode, talker mode, listener mode, or combinations of these. The system controller bit, the master clear bit, and couple of other handshake bits are also included in this register.

The interrupt register is split into R1R and R1W. R1R has the five interrupt bits and the three control bits from the incoming frame. R1W contains mask bits for each of the interrupt bits and is also loaded with the control bits of a frame to be sourced from this device.

Register 2, the data register, is also split. The read half contains the eight data bits of the incoming frame, while the write half is loaded with the eight data bits of the frame to be sourced. Writing to register 2 also initiates the transmission

FIGURE 3.1: CHIP REGISTER MAP

	BUS7	BUS6	BUS5	BUS4	BUS3	BUS2	BUS1	BUS0
Register 0 - Status Register								
Read	SC	CA	TA	LA	SSRQ	RFCR / SLRDY	CLIFCR	MCL
Write								
Register 1 - Interrupt Register								
Read	C3IN	C2IN	C1IN	IFCR	SRQR	FRAV	FRNS	ORAV
Write	C3OUT	C2OUT	C1OUT	Interrupt Enable Bits				
Register 2 - Data Register								
Read	D8IN	D7IN	D6IN	D5IN	D4IN	D3IN	D2IN	D1IN
Write	D8OUT	D7OUT	D6OUT	D5OUT	D4OUT	D3OUT	D2OUT	D1OUT
Register 3 - Parallel Poll Register								
Read	ORE	RERR	PPST	PPEN	PPPOL	P3	P2	P1
Write	-	-						
Register 4 - Loop Address Register								
R/W	Scratchpad Bits			ADD4	ADD3	ADD2	ADD1	ADD0
Register 5 - Scratchpad Register								
R/W	Scratchpad Bits							
Register 6 - Scratchpad Register								
R/W	Scratchpad Bits							
Register 7 - Auxiliary Input Register								
Read	AUX7	AUX6	AUX5	AUX4	AUX3	AUX2	AUX1	AUX0
Write	-	-	-	-	-	-	-	OSCDIS

of the frame on the loop.

The parallel poll register is register 3. It contains six bits which tell the chip how to respond when the IDY message is received on the loop. The other two bits are miscellaneous handshake bits.

Register 4 is the loop address register. It has the five bit loop address of this device and three additional scratchpad bits.

Registers 5 and 6 are both scratchpad registers available to the programmer to use as he sees fit.

TABLE 3.1: REGISTER BIT MNEMONICS

Register 0

SC - System Controller
CA - Controller Active
TA - Talker Active
LA - Listener Active
SSRQ - Send Service Request
RFCR - RFC Message Received
MCL - Chip Master Clear
SLRDY - Set Local Ready
CLIFCR - Clear IFCR Bit

Register 1

C3IN-C1IN - Control bits from received frame
IFCR - IFC Message Received
FRAV - Frame Available
ORAV - Output Register Available
C3OUT-C1OUT - Control bits for transmitted frame
SRQR - SRQ Message Received
FRNS - Frame Received Not As Sent

Register 2

D8IN-D1IN - Data bits from received frame
D8OUT-D1OUT - Data bits for transmitted frame

Register 3

ORE - Output Register Empty
PPST - Parallel Poll Status
PPPOL - Parallel Poll Polarity
RERR - Receiver Error
PPEN - Parallel Poll Enable
P3-P1 - Parallel Poll Bit Designation

Register 4

ADD4-ADD0 - Device's Loop Address

Register 7

OSCDIS - Oscillator Disable
AUX0-AUX7 - Flag input lines

Register 7 is the auxiliary input register. The read-only part of this register contains the current state of the eight input flag lines. The write-only portion only has one bit, the oscillator disable bit, whereby the programmer can turn off the chip oscillator for minimum power consumption.

In the descriptions which follow, some confusion could arise between mnemonics used for the chip and mnemonics used to refer to HP-IL interface function states. To avoid this, interface function mnemonics will always be enclosed in braces {}. {AIDS} refers to the interface function state while AIDS refers to a state of a state machine in the chip, which may not correspond to the interface function at all.

3.2 The Status Register (R0)

The device status on the loop is set by the state of the four most significant bits of register 0. The device attributes of system controller, controller, talker, and listener are set with their corresponding bits. With one minor exception, these bits are controlled by writes from the CPU, so that it is largely the responsibility of the firmware to decode the appropriate frames and set the proper bit(s). These status bits then cause the chip to automatically decode most subsequent frames so that only those frames which affect the device are routed to the CPU while others are automatically retransmitted.

The other bits in R0 provide some miscellaneous capabilities which include sending service request, handling the CMD-RFC handshake, resetting the IFCR bit, and initializing the chip.

The System Controller (SC) Bit

Whenever the MCL (Master Clear) bit is true (1), the SC bit is set to the state of the $\overline{\text{SCTL}}$ pin. When MCL is false, SC may then be set and cleared by the CPU. SC controls the decoding of the IFC message. If SC is true, ■ incoming IFC is presumed to have been sourced by this device, and will be error-checked. If SC is clear, the IFC must be from some other device (the system controller) and the IFCR interrupt bit will be set accordingly.

The Controller Active (CA) Bit

This bit can only be set and cleared by the CPU. It should be true whenever either the {CACS} interface state or the {CSBS} interface state is active. When CA is true, that class of messages which is normally sourced by the active controller (CMD, IDY, and some RDY messages) will be error-checked when they are received. Otherwise, the disposition of these frames is dependent on the TA and LA bits.

The Talker Active (TA) Bit

This bit is set and cleared by the CPU. It should be true whenever any of the interface states {TADS}, {TACS}, {SPAS}, {DIAS}, or {AIAS} is true. When TA is true, data (DOE) frames

are error-checked. If TA is false, data frames are handled according to the state of the LA bit.

The Listener Active (LA) Bit

This bit is set and cleared by the CPU. It should be true whenever either the {LADS} interface state or the {LACS} interface state is true. If LA is true, incoming data frames are passed to the CPU via an interrupt. If LA is not true, data frames are automatically retransmitted (assuming TA is also not true).

In an idle device, all four status bits will be false. In this condition most frames are retransmitted without intervention from the CPU. Often more than one of the bits will be true at the same time. For example, if the active controller is also the active talker (a common situation), both CA and TA must be set.

It is not particularly useful to have a device be a talker and a listener at the same time, so that combination (TA and LA both true) is reserved to perform a special function. In this condition all frames are routed to the CPU with an interrupt; no frames are automatically retransmitted by the chip. This permits the device to serve as a loop analyzer, monitoring all activity on the loop. The TA.LA mode is also used in certain situations, such as asynchronous loop operation, where the normal loop handshake needs to be disabled or bypassed. While this mode of operation can be very useful, it is important to remember that because no frame is automatically retransmitted, loop speed will slow down substantially. If the loop analyzer is also the active controller, CA must also be set.

Complete information on the interaction of the status bits, the incoming frame, and the various interrupts is given in the following sections.

The Send Service Request (SSRQ) Bit

This bit is set and cleared by the CPU. If CA is true, SSRQ has no effect. In devices which are not the active controller, SSRQ set will cause the SRQ bit to be set in appropriate frames. This occurs both for DOE and IDY frames automatically retransmitted by the chip and for DOE frames sourced or retransmitted by a CPU write to R2. The state of the SRQ bit in the chip input and output registers is not modified, only the value of the outgoing bit.

The RFC Received (RFCR) Bit

RFCR is a read-only bit which can be ignored in most common applications. In analyzer mode (TA.LA=1) and in controller mode (CA=1) this bit will always read false (0). In other modes, RFCR will be set when an RFC frame is received. The bit will return to its normal false state when the CPU writes the SLRDY bit true (enabling automatic retransmission of the RFC) or when an IDY or CMD frame is received (destroying the RFC; see section on asynchronous loop operation), whichever occurs first.

The Set Local Ready (SLRDY) Bit

This write-only bit provides the chip with a simple means of handling the CMD-RFC handshake. When a CMD is received which affects this device, the ensuing RFC will not be retransmitted until the CPU indicates that it is ready to receive the next frame by writing SLRDY true. The means of retransmission of the RFC is invisible to the CPU; no other action should be taken beyond writing SLRDY true. If a CMD is received which does not affect this device, the CPU is not interrupted. In this case the SLRDY bit is set automatically by the chip without the need for any CPU action. This bit is self-resetting so that the CPU never needs to clear it. If an IFC command is received, the IFCR bit must first be cleared (by writing CLIFCR true) before SLRDY can be set to enable RFC transmission. Complete data on decoding of commands is given in the following section.

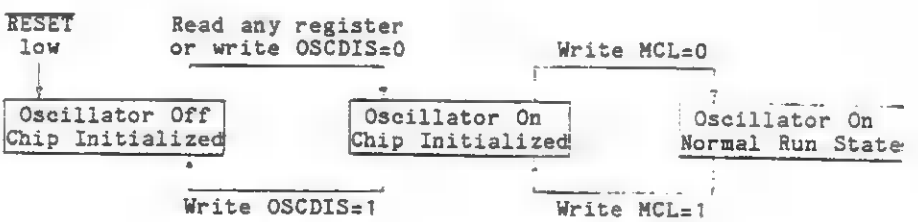
The Clear IFCR (CLIFCR) Bit

The CPU writes this bit true to clear the IFCR interrupt bit. This is the only way to clear IFCR. CLIFCR is self-resetting. It will return to its normal false state one to two microseconds after it has been set. Reading CLIFCR will always return a zero except in that brief interval immediately after it has been set true.

The Master Clear (MCL) Bit

This bit is set by the CPU when it is necessary to initialize the chip. If the RESET pin is driven low, the oscillator is shut off and MCL is set automatically. To return the chip to the normal run state, the oscillator is first turned on by reading any register or by writing the OSCDIS bit false, and then the MCL bit is written false. The following state diagram summarizes this:

FIGURE 3.2: CHIP INITIALIZATION



Chip initialization puts the chip in an initial state defined as follows:

1. Interrupt bits IFCR, SRQR, FRNS, and FRAV are cleared.

2. The interrupt bit ORAV is set high.

3. The five interrupt enable bits are cleared (No interrupt will occur until one or more of these bits is set high by the CPU).

4. The SC bit is set to the state of the $\overline{SCTL}$ pin.

5. The Retransmit Service Request latch (RTSR) is cleared. RTSR will be described later in this document.

6. The ORE bit is set high.

3.3 Frame Decoding

It is very important for the user to understand to what extent received frames are decoded by the chip and what decoding must be performed by the program in the CPU. The setting and resetting of the interrupt flags in register 1 is directly affected by the decoding of the incoming frame and the chip status contained in register 0. The transmitter logic is also controlled by the received frame and the device status. The chip may automatically retransmit the received frame, or it may source an RFC, or it may do neither, in which case the CPU must send the frame if retransmission is necessary.

Data Frames

All DOE messages are decoded in one group. DAB and END frames are treated identically. If the device is the active talker (TA true), these frames are automatically error-checked. The active listener (LA true) passes these messages to the CPU for interpretation; retransmission is done by the CPU. In all other devices (TA and LA both false), DOE messages are automatically retransmitted by the chip.

Identify Frames

IDY messages are also decoded in one group. The active controller (CA true) automatically error-checks these frames; all other devices automatically retransmit them.

Command Frames

Command frames are automatically retransmitted by all devices except the active controller. When CA=1, CMD frames are decoded into two groups. The first group consists of only the IFC command and the second is comprised of all other commands. If the controller has SC=1, both groups are error-checked when received, and if there is no error, the RFC is automatically sourced. In controllers which are not the system controller, the second group is handled as above, but the IFC is passed to the CPU and is not retransmitted. This must be done by the program in the CPU by writing the IFC to the output register.

As mentioned previously, all commands are automatically retransmitted in non-controller devices (CA=0). Some of these will affect the device, and therefore are passed to the CPU as well. In these devices, CMD messages are decoded into three groups. Once again, the IFC frame is in a group by itself. The second class consists of commands other than IFC which may affect this device. This group is called the Command Interrupt group (CMDI). The last group contains all commands not in the other two groups. These are not passed to the CPU but are automatically retransmitted (as are the other two groups).

The classes of commands which fall in the CMDI group (and consequently those which fall in the third group) depend on the device status. If the device is either Talker or Listener Active, all commands (except IFC) are in the CMDI group. In ■ device with both LA and TA false, the CMDI class consists of universal commands (UCG), secondary address commands (SAG), and talker and listener address commands (TAG, LAG) if the address in the command matches the device's loop address contained in the low order five bits of register 4. The following expression summarizes this relationship:

$$CMDI = \overline{CA}.CMD.\overline{IFC}.(LA+TA+UCG+SAG+(TAG+LAG).(\text{addresses equal}))$$

In summary then, for devices with CA=0, the IFC command is automatically retransmitted and passed to the CPU, the CMDI group is automatically retransmitted and passed to the CPU, and all other commands are automatically retransmitted but do not interrupt the CPU.

Ready Frames

RDY messages also break down into three classes: the RFC message, the Addressed Ready Group (ARG) frames, and the Auto Address Group (AAG) frames.

In the active controller, the RFC message is automatically sent by the chip when the previously issued command returns and error-checks OK. When this RFC returns, the controller is simply notified that the loop handshake process is complete so that the next frame can be sourced. When ■ non-controller device receives an RFC, it is automatically retransmitted as soon as the SLRDY bit is set high (SLRDY is set either automatically or by the CPU, depending on the received command).

In idle devices (LA+TA+CA=0), ARG frames are automatically retransmitted by the chip. In active devices (LA+TA+CA=1), these messages are passed to the CPU for interpretation. Retransmission is accomplished by the CPU writing the frame to the output register. While there are currently no ARG frames that listeners repond to, this condition is included for possible future expansion of the protocol.

AAG messages are passed to the CPU of all non-controller devices (CA=0). The CPU must write the frame to the output register in order to retransmit it. In the active controller, the AAG group is error-checked, since these frames are sourced by this device.

It is important to remember that in analyzer mode (TA.LA=1) no frame decoding, no error-checking, and no automatic retransmission is performed. The frame is merely passed to the CPU and it is the responsibility of the program to perform these functions.

3.4 The Interrupt Register (R1)

The interrupt register contains not only the five interrupt flags and their corresponding enable bits, but also the three control bits from the received frame to be passed to the CPU and the three control bits for the frame to be transmitted by the CPU. Note that the chip does not have to be used in an interrupt scheme. The enable bits could all be cleared (0) to prevent any interrupt from occurring; the interrupt flag bits function in the same manner regardless of the state of the enable bits so that the CPU could simply read the interrupt bits in a loop in the program and handle them as they are set by the received frames from the loop.

The Control Bits

The read-only bits C3IN-C1IN are loaded by the chip with the control bits from the incoming frame. These bits are then held until the CPU reads the received frame from register 2, at which time the control bits from the following frame are loaded into C3IN-C1IN. For this reason it is important for the CPU to read the control bits from register 1 before reading the data bits from register 2 to preclude mixing of the data and control bits from different frames.

C3OUT-C1OUT are write-only bits which the CPU loads with the control bits for the frame to be transmitted, prior to writing the data bits to register 2, which initiates the transmission. It is often necessary for the CPU to retransmit the identical frame which it received, so to make this more convenient, C3OUT-C1OUT are automatically loaded with the contents of C3IN-C1IN when the CPU reads the received frame from register 2. In this way, the CPU only needs to write the data bits to register 2 and does not need to worry about the control bits in R1W if it wishes to retransmit the frame without modification.

The IFC Received (IFCR) Bit

This bit is set whenever an IFC command is received, regardless of the setting of the status bits. IFCR is cleared when the CPU writes a logic one in the CLIFCR bit in register 0 (it is not necessary to clear CLIFCR as it is self-resetting). The chip master clear (MCL=1) also resets IFCR.

The Service Request Received (SRQR) Bit

If the chip is controller active (CA=1) and a frame arrives with a service request (C1=1 in DOE or IDY frames), the SRQR bit is set. If a data or identify frame without the service request bit set arrives, the SRQR bit is cleared. Master clear also

resets the SRQR bit. If CA=0, the SRQR bit is always zero.

The Frame Available (FRAV) Bit

In general, when a frame is received which affects the device, the CPU is notified by setting the Frame Available bit and generating an interrupt if the corresponding enable bit is set (1). Specifically, FRAV is set under the following specific conditions:

1. LA=1 and a data frame (DOE) is received.
2. TA+LA+CA=1 and an ARG ready frame is received.
3. CA=0 and a CMDI command or AAG ready frame is received.
4. TA.LA=1 and any frame is received (analyzer mode).

When the CPU reads register 2 to obtain the frame's data bits, FRAV is reset to zero. Remember that the frame's control bits from register 1 should be read first. MCL=1 also resets FRAV.

The Frame Received Not As Sent (FRNS) Bit

When a frame is received that is automatically error-checked by the chip and an error is detected, the frame is loaded into R1 and R2 and the FRNS bit is set. Note that FRAV and FRNS are mutually exclusive. As with FRAV, FRNS is reset when the CPU reads register 2. Likewise, master clear also resets FRNS.

The Output Register Available (ORAV) Bit

The most important use of ORAV is to indicate to the sourcing device (either talker or controller) that the loop handshake is complete and the next frame may be sent. There are several cases, however, and they do not all fit this description, so they will be explained individually.

Following completion of error-checking, ORAV is set. This is the case mentioned above. The only exception is when a CMD returns to the active controller and error-checks correctly. In this situation, an RFC is automatically sourced and ORAV is not set until the RFC returns.

ORAV is set upon receiving a frame which will subsequently be written to the output register for retransmission. These are the same frames listed as setting the FRAV bit except that CMDI frames do not set ORAV since they are retransmitted automatically by the chip.

If the active controller is not the system controller and an IFC is received, ORAV is set (along with IFCR). In this case, the command (IFC) is not automatically retransmitted. After responding to the IFC, the CPU must write it to the output register for retransmission.

Anytime the device is not the active controller and is also not the active talker (TA and CA both zero), ORAV will remain high regardless of the received frame and regardless of CPU writes to register 2. ORAV will normally by masked out by setting its enable bit to zero in this situation. Note also that

master clear (MCL=1) sets ORAV high.

Except for the case of CA and TA both zero, ORAV will be reset to zero by writing the frame to be sourced or retransmitted to the output register.

Whenever one or more of the interrupt bits together with the corresponding enable bit is high, the $\overline{IRQ}$ interrupt line to the CPU will be low. The CPU will respond, clearing or masking the bit, and $\overline{IRQ}$ will return to its normal high state. Unless the interrupt flag bit is masked or cleared, no further interrupts can occur since the interrupt line will remain in its low state.

3.5 Frame Processing Summary

The interaction of the incoming frame, the chip status bits, and the interrupt flag bits is fairly complex. An attempt is made in this section to bring all the pertinent information together in one place and to tabulate it in a manner so that it is clear, concise, and easy to use.

Transmitted Frames

No circuitry is provided within the chip to prevent the transmission of improper frames. Thus, it is the programmer's responsibility to see that the frames which are transmitted do indeed conform to HP-IL protocol. Unless this precaution is carefully observed, it is relatively easy to cause a situation in which a frame or frames circulate endlessly or a situation in which the loop is "hung" or "crashed" and will not respond.

In general, idle devices and listeners do not source frames, they only retransmit frames which they receive. The only exception is if a device is able and is enabled to generate asynchronous requests. Under this condition only, an idle device or an active listener may source an IDY with the SRQ bit set. A change in protocol is currently under consideration which would allow the active listener (LA=1) to source the NRD message when able and enabled by the controller.

The active talker may source data frames and the EOT messages (TA=1). Note that even though the TA bit is set after the talk address command is received, sourcing of frames may not begin until the appropriate SOT message is received. Talkers may also source the asynchronous request message under the same conditions described above.

The active controller (CA=1) may source commands (but not IFC unless SC=1 also), ready frames (except EOT), and IDY frames. Devices which are both the active controller and the active talker or listener take the attributes of both functions.

In analyzer mode (TA.LA=1) the CPU must keep track of the function currently enabled and only source the proper frames. The device could be simply a repeater, an idle device, a listener, the talker, or even the controller (provided that CA=1

also).

Once again, unless the chip status bits are set properly for the frame being transmitted and unless proper HP-IL protocol is observed by the CPU program, the integrity of the entire interface system will be jeopardized.

Received Frames

A thorough understanding of the table at the end of this section is critical to the correct programming of a CPU using this HP-IL interface chip. Several explanatory remarks are necessary prior to understanding the table. These remarks, together with the table provide the information necessary to properly process incoming frames.

It is important to remember that at all times a device must obey the interface function state diagrams as defined in the HP-IL defining document "Hewlett-Packard Interface Loop". The designer must have a full understanding of the state diagrams for the interface functions used in his device prior to programming the CPU. It is strongly recommended that these state diagrams be followed exactly even when frames are received which do not obey correct protocol.

The table on the following page gives the interrupt flag response for various combinations of chip status and received frames. For clarity and space reasons, certain of these combinations are described in the text rather than in the table.

The IFCR flag is set anytime the IFC command is received, regardless of the device status, even if this device sourced the IFC. To clear IFCR, the CPU must write the CLIFCR bit high (MCL=1 also clears IFCR). Most devices will clear IFCR, do whatever is necessary to execute the IFC, and then write SLRDY high to enable automatic retransmission of the ensuing RFC. When an active controller (not the system controller) receives an IFC, it must first set CA=0 and then retransmit the IFC by writing it to register 2. It can then respond as any other device, by writing CLIFCR, executing the IFC, and writing SLRDY. When the system controller (SC=CA=1) receives its IFC back, it should merely clear IFCR and then wait for the automatically sourced RFC to return, which will set ORAV.

The SRQR flag is only used by the active controller. SRQR will always be zero as long as CA=0. If CA=1 and a DOE or IDY message is received with the SRQ bit set, SRQR will be set high. SRQR will remain high until a DOE or IDY is received with a zero SRQ bit, or CA is reset to zero, or master clear is set high.

Because of the logic in the chip, anytime both CA and TA are zero the ORAV flag will be high. This is true regardless of the received frame. For interrupt operation in this condition, the ORAV bit must be masked by setting its enable bit to zero.

Analyzer mode is also not shown in the table. When both TA and LA are high, any received frame will set FRAV and ORAV. No frame decoding, automatic retransmission, or error-checking is

performed. The CPU must take care of these functions in this mode. The operation of the IFCR and SRQR flags is not affected, however.

TABLE 3.2: INTERRUPT FLAG RESPONSE

Received Frame	Chip Status SC	CA	TA	LA	FRAV	FRNS	ORAV	Notes
DOE	X	X	0	0	-	-	-	Auto Rtrn
	X	0	0	1	S	-	-	CPU Rtrn
	X	X	1	0	-	E	S	Src, Auto Err Chk
	X	1	0	1	S	-	S	CPU Rtrn
IFC	0	0	X	X	-	-	-	Auto Rtrn
	0	1	X	X	-	-	S	CPU Rtrn
	1	1	X	X	-	E	E	Src, Auto Err Chk, Auto Src RFC If OK
CMDI	X	0	X	X	S	-	-	Auto Rtrn
CMD.$\overline{\text{IFC}}$.$\overline{\text{CMDI}}$	X	0	0	0	-	-	-	Auto Rtrn, Auto SLRDY
	X	1	X	X	-	E	E	Src, Auto Err Chk, Auto Src RFC If OK
RFC	X	0	X	X	-	-	-	Auto Rtrn After SLRDY
	X	1	X	X	-	-	S	Src, Decoded
ARG	X	0	0	0	-	-	-	Auto Rtrn
	CA+TA+LA=1				S	-	S	CPU Rtrn or Src (No Auto Err Chk)
AAG	X	0	X	X	S	-	S	CPU Rtrn
	X	1	X	X	-	E	S	Src, Auto Err Chk
IDY	X	0	X	X	-	-	-	Auto Rtrn
	X	1	X	X	-	E	S	Src, Auto Err Chk

X Don't Care. Some combinations are explained in the text, so the table does not necessarily represent all possible don't care states.

S The bit is set high when the specfied frame is received and the chip status is as shown.

E The bit is set high only if automatic error-checking detects an error.

- This combination has no effect on this bit.

The table also indicates other actions taken by the chip when certain frames are received and chip status is in a certain state. The chip may automatically retransmit the received frame (Auto Rtrn) or the CPU may be required to retransmit the frame by writing it to register 2 (CPU Rtrn). The chip may have been the source of the received frame (Src) and in this case, the received frame may be automatically error-checked (Auto Err Chk) to see that no transmission error occurred. When a controller receives a command which it sourced, and the command error-checks OK, the chip will automatically source the RFC message (Auto Src RFC If OK). When an idle device receives a command which will not affect it, it automatically sets the SLRDY bit to enable retransmission of the ensuing RFC without CPU intervention (Auto SLRDY).

Remember that FRAV and FRNS are mutually exclusive and that they are normally reset by reading register 2. It will usually be necessary to read the control bits from register 1 first. Setting master clear also resets FRAV or FRNS. Provided CA and TA are not both high, ORAV is reset by writing the frame to be sourced or retransmitted to register 2. ORAV is set high by master clear.

3.6 The Other Registers

The operation and usage of the remaining registers is fairly straightforward and they will be explained in the various parts of this section.

The Data Register (R2)

When FRAV or FRNS is set, the received frame is loaded into R1R and R2R. The CPU may then read the received frame by reading first R1 and then R2. The act of reading R2 resets FRAV or FRNS, whichever was set, and allows the next received frame to be loaded into R1 and R2. This is why it is important to read R1 first. Otherwise, the control bits might be overwritten with the control bits from the following frame before they could be read. Note also that reading R2 loads the control bits from R1R to R1W in preparation for retransmission of the received frame.

In retransmitting a received frame, the CPU only needs to write the data bits to R2W, since the control bits will already have been correctly loaded (by the read of register 2) into R1. When the device needs to source a frame, it should first set up the correct control bits by writing to R1, and then write the data bits to R2. In either case, when R2 is written, the chip automatically loads the control bits from R1W and the new bits from R2W into the output register and begins to transmit them on the loop. If the transmitter is busy with a frame which is being automatically retransmitted, the transmission of the frame from register 2 will simply be delayed until the transmitter is idle.

The Parallel Poll Register (R3)

The ORE (Output Register Empty) bit is used to monitor the

transmission of frames from register 2 to the loop. This bit is normally high. The write to R2W sets ORE low; it will return high again when all eleven bits have been transmitted. This bit only needs to be monitored if the possibility exists that R2 might be written to again before all bits have been transmitted. Frame transmission time is nominally 46 microseconds, but longer delays will occur if the transmitter is already busy when the write to R2 takes place.

The RERR (Receiver Error) bit is normally low. If a sync bit is received in the middle of a frame, it is set. It is reset by a write to register 3. This bit is usually ignored. The chip automatic error-checking capability will catch these kinds of problems.

The six low order bits of register 3 control the chip's response to parallel polls conducted by the active controller (IDY messages). The four low order bits should be loaded with the four low order bits of the PPE command. They cause the chip to respond on the correct bit and in the proper polarity. The PPEN (Parallel Poll Enable) bit should be set by the CPU when the PPE command is received. It should be reset if the PPD message is received and the device is listener active, or if the PPU command is received. The PPST (Parallel Poll Status) bit should be set by the CPU when it wishes the device to respond affirmatively to the parallel poll. This bit is analogous to the SSRQ bit in register 0. Note that both PPEN and PPST must be high before the chip will automatically respond to parallel poll. All six of these bits are set and reset by the CPU only.

The Loop Address Register (R4)

The three high order bits of register 4 are scratchpad bits which the programmer may use. The low order five bits should be loaded by the CPU with the loop address of this device. If the device is idle (CA=TA=LA=0), and a talker or listener address command is received in which the address bits match the bits in register 4, FRAV will be set and the command will be passed to the CPU. Note that UNT (TAD 31) and UNL (LAD 31) are not treated any differently than any other TAD or LAD message. This could be important if the CPU ever stores address code 31 in register 4. All bits in this register are set and reset by the CPU only.

The Scratchpad Registers (R5, R6)

These two registers are provided entirely for the use of the programmer. The bits in these two registers are controlled only by writes from the CPU.

The Auxiliary Input Register (R7)

The read-only bits of register 7 continuously reflect the state of the eight auxiliary input flags. They may be read by the CPU at any time. If they are not connected externally, they will read high. There is only one bit in the write-only side of register 7, the OSCDIS bit. Its use was descibed together with the MCL bit of register 0. Note that OSCDIS has no effect on the oscillator unless MCL is high.

3-7 Some Additional Notes

The Retransmit Service Request (RTSR) Latch

Most of the time, if the SRQ bit is set in a data frame, it will be automatically retransmitted around to the controller. If the controller is downstream of the talker but upstream of the device requesting service, however, a problem would exist. The talker's control bits in R1W will normally be all zeroes (data frames), so that even if the SRQ bit was set on a returning data frame, when the talker writes the next frame, that bit will be zero. Thus the service request would not reach the controller. To prevent this from happening, the chip has a additional latch, RTSR, which remembers that the SRQ bit was set on an incoming data frame and sets the bit on the next transmitted data frame without changing the control bits in R1W.

RTSR is set when TA=1 and a data frame is received with the SRQ bit set high. The latch is cleared when the talker has sourced its next data frame with the SRQ bit set high or when a frame has been automatically retransmitted which has bit C1 low. This could be an IDY (indicating that the device is no longer requesting service) or it could be a command from the controller (this should not occur as it is not correct protocol). Master clear also resets RTSR. The programmer can ignore RTSR, as its operation is completely automatic.

It should also be mentioned in connection with service request that the controller has logic which "kills" the SRQ bit (both in data frames which it automatically retransmits and in IDY frames which it sources) so that SRQ will not circulate endlessly.

Received Frame Buffering and Transmission

As a frame is received, it is loaded into a circuit element called the input buffer. From here the frame is loaded into the input register and from the input register to R1R and R2R. One can see therefore, that at most the chip is capable of holding three frames before data is lost. If the input register is empty when a frame is received, it is loaded directly through the input buffer to the input register bit by bit as it is received. If the input register is already full, the frame will be held in the input buffer until the input register is available and will then be loaded in parallel to the input register. When all eleven bits have been received, the frame is loaded from the input register to R1R and R2R. If the previous frame has not been read from R2R, this load will be delayed until the CPU reads R2. The read of R2 will reset some of the interrupt flags and permit the frame in the input register to be loaded into R1R and R2R and set the proper interrupt flags for this new frame, depending on the device status and the received frame.

Of course, not all frames will be loaded into R1R and R2R. Depending on the device status, certain frames will be automatically retransmitted and will not interrupt the CPU by being loaded into R1R and R2R. This automatic retransmission is done from the input register. Depending on the received frame,

and assuming the input register is empty, this retransmission may start after the first bit is received, or after the fourth bit, or after the sixth bit. If the input register is busy when the frame is received, the retransmission will be delayed until the register is available or enough bits have been received, whichever occurs later. As the automatic retransmission is taking place, the chip logic may insert the SRQ bit or the parallel poll response bit or both, depending on the frame and the state of the SSRQ bit and the parallel poll bits in register 3.

Except in analyzer mode, the IFC and RFC frames are never loaded into R1R and R2R. The IFC frame always sets the interrupt flag and may or not be retransmitted, depending on device status, but will never go into R1R and R2R. If SLRDY is true while the RFC is being received, it will be retransmitted from the input register as are other frames. If SLRDY goes true after the RFC is received, the chip will generate its own RFC and transmit it (the RFC encoder). If an IDY or CMD is received while holding an RFC and waiting for SLRDY, the RFC will be destroyed. Because of the different ways in which RFC can be decoded (somtimes only the first six bits are looked at, sometimes all eleven are necessary), no other frame may be the same in the first six bits as an RFC, or equivalently, the last five bits of an RFC must always be zero.

If the $\overline{\text{RTRN}}$ line is pulled low externally, the chip will act as a simple repeater (provided, of course, that it remains powered and the oscillator is on). All frames will be automatically retransmitted as soon as they enter the input register.

It is seen from the preceeding discussion that the chip transmits frames from three sources: the output register which consists of R2W and the three extension bits loaded from R1W, the input register for automatic retransmission, and the RFC encoder. Chip logic selects the proper source at any given time. It should also be mentioned that the chip includes logic to prevent the transmitter from overrunning the receiver when automatically retransmitting.

Automatic Error-checking

In general, when the chip sources a frame and the frame returns, the received frame is compared with the transmitted frame and any difference is reported to the CPU. The received frame is compared bit by bit as it enters the input register with the contents of the output register (R2W plus the three extension bits loaded from R1W). Clearly, if the CPU writes to the output register again before the frame returns, the error-checking function cannot work correctly. Since the three control bits determine the type of frame and consequently whether or not error-checking should even be performed, they are assumed to be correct. Some errors in the control bits will still be caught, but it is possible for noise to create a frame which will circulate endlessly or "hang" the loop if the control bits are modified. All eleven bits of the received frame are compared, except that the SRQ bit will not cause error-checking to report an error in those frames which can have the SRQ bit. SRQR will

still be set in the device which has CA=1 and RTSR will still be set in the active talker when these actions are appropriate. When an error is detected, the frame is loaded into R1R and R2R and FRNS and ORAV are set. FRNS is set before ORAV to prevent the CPU from accidentally catching ORAV but not FRNS if it reads the bits at the moment they are being set.

In some situations, FRNS does not really indicate an error. When the talker receives its last data frame after an NRD sequence FRNS will be set, since the NRD is in the output register and will certainly not match the data frame. It is recommended that the CPU error-check this last frame in software and source ETO or ETE accordingly. Also, if parallel poll is enabled, the controller's IDY will not error-check correctly if one or more of the devices responds to the poll and modifies the data bit(s) in the frame. FRNS is used in this case to indicate that there was a response to the controller's poll.

If a frame is written to the output register while error-checking is in progress, the error-checking operation is aborted without affecting FRNS and ORAV. This should never occur when the program is running correctly.

Asynchronous Operation

Normally, only a single frame is on the loop at any time. A new frame is not transmitted until the first frame arrives at its destination, which is usually the sourcing device, and has been processed. This is referred to as synchronous operation. There are, however, a very few situations where asynchronous sourcing of frames is permitted. In these cases there can be more than one frame on the loop simultaneously. Because of the difficulty in handling these cases, this type of operation is very carefully defined and restricted. The device designer and programmer must be thoroughly familiar with section 5.4, "Asynchronous Loop Operation", of the document "Hewlett-Packard Interface Loop" in which the allowable asynchronous operations are defined. Asynchronous operation primarily concerns devices which can be controllers. When a device is not the active controller it generally need not concern itself with the considerations discussed in this section.

Asynchronous operation is a problem for the chip because of two reasons. The frame which is sourced may return either before or after a previously sourced frame (from this device or from another device) so that error-checking will not function correctly. In addition, it is an unfortunate characteristic of the chip that when an RFC has been received and is waiting for SLRDY, and an IDY or CMD is received (asynchronously), the RFC is destroyed. The recommended method for dealing with these problems is as follows: prior to performing any asynchronous operation the controller is placed in analyzer mode (TA=LA=1). In this way the chip performs no automatic functions but merely passes all frames to the CPU for interpretation. The CPU can then perform manual error-checking and determine exactly what is happening on the loop at all times during the asynchronous sequence. Specifically, the controller should set TA and LA and then wait 50 microseconds to permit any frame currently being

received to be processed and set the interrupt flags before sourcing the asynchronous frame. After checking the flags the controller can then go ahead and source the asynchronous frame. After finishing the operation and making sure that no more than one frame is still on the loop, the controller can return to its normal mode by resetting the proper status bits. After waiting another 50 microseconds (if there was one more frame yet on the loop) and checking the flags, the controller can continue normal operations.

Suppose, for example, that the system controller has passed control to another controller and now decides to take control of the loop by asynchronously sourcing an IFC command. The CPU would set TA=LA=1 and then wait as described above. If the flags then indicate no frame, the CPU would set CA=1 and send the IFC. The CPU should monitor FRAV and IFCR awaiting the return of the IFC. If other frames are received, they should be read and ignored. When IFCR indicates that the IFC has returned (error-checking is not necessary here since the frame is decoded), the CPU should send the RFC (the chip will not source it automatically in this mode). The CPU should now wait for the RFC, reading and ignoring any other frames. When the RFC returns, the system controller can reset TA and LA and resume normal control of the loop. It is possible that the asynchronous IFC could destroy some other frame on the loop, but because of the nature of this command, this is unimportant.

For a second example, assume that a slow talker and listener are transmitting data and the controller needs to determine if any devices require service more often than the slow data frames allow this information to reach the controller. Or assume that the controller simply gets worried and wants to verify quickly that the loop is not broken. In either case, the controller will want to send an asynchronous IDY. The controller's CPU should set TA and LA, wait the specified interval, check the flags, and then send the IDY. When FRAV and ORAV are then set by a received frame, the CPU will read R1R and R2R and find, say, a data frame. The CPU will retransmit this frame by writing R2W. The next frame will then be the IDY. The CPU reads this and does error-checking in software. The CPU can now reset TA and LA, check the flags (the following frame could have been received just before the flags were reset), and the loop is back in normal operation.

In the second example, it is very important that the IDY not destroy the other frames on the loop and the chip is designed so that this is accomplished. The buffering of received frames is such that the IDY will be automatically retransmitted without disturbing the frame in R1R and R2R. Unfortunately, there is one situation where the asynchronous IDY does destroy the frame. The controller can source asynchronous IDY frames not only during data transmissions but also during its own frame transmissions (CMD, RDY frames). Because of an anomaly in chip design, if an IDY overtakes an RFC, the RFC is destroyed. The controller can avoid this situation by simply not sending an asynchronous IDY immediately after the RFC (the controller has no way to detect when the RFC is transmitted unless it is already in analyzer mode when the preceeding command is sent, and thus sends the RFC manually). If the controller must send the IDY after an RFC, it

should send a second RFC if the IDY returns first, in order to complete the CMD-RFC handshake sequence. Controllers need to remember, however, that the time will come when HP-IL devices may use some other chip which does not have this problem. Sending a second RFC in that situation may result in both RFC's coming back at different times. The controller will have to be absolutely sure that the handshake is complete and that there are no more RFC frames on the loop before returning to normal operation. Similar problems arise when the controller enables the asynchronous request mode in which other devices can source their own asynchronous IDY frames. The controller must always make sure the loop is "flushed" before resuming normal operation.

4. THEORY OF OPERATION

All necessary information to use the chip in a device has been given in previous chapters. For a complete understanding of the way in which the chip performs its functions some additional information is presented in this chapter, however.

The block diagram of the chip circuitry on the following page shows only the major circuits and control and data lines for clarity. The HP-IL input is in the upper left corner while the output is in the upper right. The general purpose microprocessor bus interface is in the lower right. The chip oscillator lines and miscellaneous control lines are along the left edge and lower left.

The 2 MHz oscillator generates a 2 MHz clock for the HP-IL circuitry and drives a clock generator circuit which provides a 500 kHz two-phase clock for the PLA's. The rest of the chip, consisting mainly of the microprocessor interface, is asynchronous and requires no clock signal.

The input detector and receiver control logic convert the HP-IL input signal to logic levels, reset the input pointer and clear the input buffer when a sync bit is detected, and load the succeeding bits into the input buffer via the input pointer and the input demultiplexer. Under control of the PLA, the receiver logic also causes the input register to be loaded at the proper time. The presence of an IDY or CMD message in the input buffer is decoded. No other decoding is done until the message is in the input register.

From the input register, the frame is decoded and the frame type is fed to the PLA. The frame may be loaded into R1R and R2R and the interrupt bits set, or it may be fed through the input register multiplexer for retransmission or error-checking.

The transmit encoder receives inputs from three sources. Data from the output register goes through the output register multiplexer under control of the output pointer. Data from the input register may be directed through the input register multiplexer under control of the output pointer for retransmission. The digital comparator prevents the transmitter from overrunning the receiver. The third source for data is the RFC encoder. The source of the data is selected by the driver PLA. The transmit encoder may also insert the SRQ bit or the proper parallel poll response. Finally, the data from the output register may be compared with the data from the input register for error-checking by enabling both multiplexers at the same time.

The acceptor PLA is the main control block for the chip. It receives as inputs the frame type and the status bits from R0 and

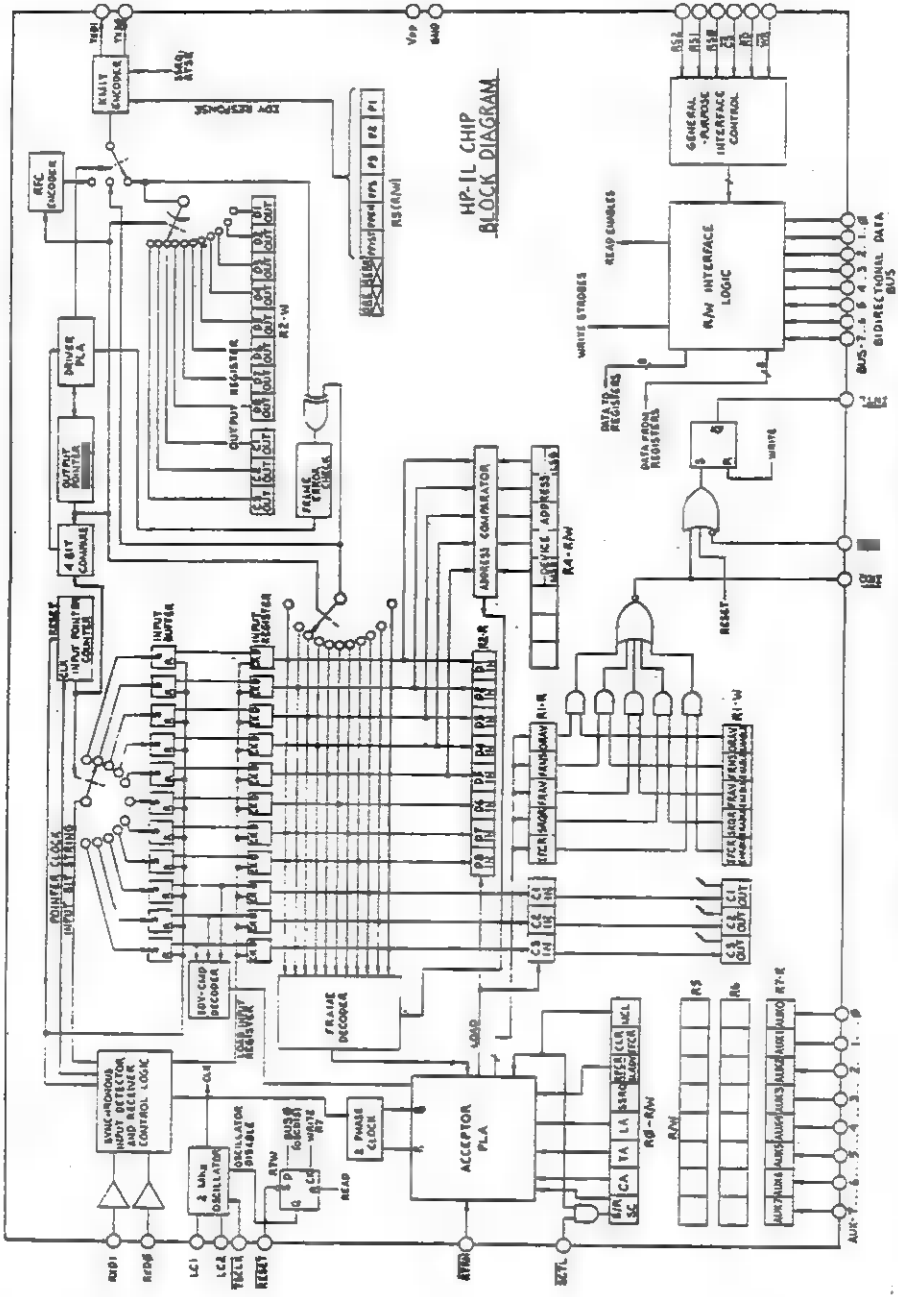


FIGURE 4.1: HP-IL CHIP BLOCK DIAGRAM

its outputs control almost all of the HP-IL portion of the chip. The driver PLA controls frame transmission.

The read/write interface logic and control merely provides access to registers 0-7 via separate internal 8-bit wide read and write busses (not shown in the block diagram for clarity). The interrupt logic signals the CPU when appropriate conditions occur.

Control of the entire chip resides in several state machines, of which the largest and most complex is the acceptor state machine. An understanding of the block diagram on the previous page and the operation of the various state machines will provide a fairly complete understanding of the chip at very nearly the circuitry level.

It is extremely important to remember that the states in the chip state machines may have the same or similar names as states in the interface functions which define the operation of the HP-IL system. Great care should be taken not to confuse the two, as they may not even perform a similar function. For clarity, states in the interface function definition will be set off by braces {}.

Before describing the detailed operation of the various state machines, it is necessary to understand the derivation of several important signals on the chip that serve as inputs to the state machines in addition to the decoding of the incoming frame.

TABLE 4.1: CHIP SIGNAL DESCRIPTION

EOL	End of load. A short pulse which occurs immediately after the last bit (D1) has been loaded from the input buffer to the input register.
EQ	Equal. This signal is set true when the acceptor is in the ALIR state and is only set false if automatic error-checking detects an error in the AECS state.
FRMR2	Frame in register 2. FRMR2 indicates that a frame has been loaded into R2R and the upper three bits of R1R. FRMR2=FRAV+FRNS.
IBIC	Input buffer contains an IDY or CMD. This is the only frame decoding done in the input buffer.
IPGE35	Input pointer greater than or equal to 3 or 5. This signal is used to enable automatic retransmission of CMD, IDY, and ARG frames. The input pointer counts the number of bits received. If the input register contains an IDY or CMD frame, the signal goes true when the input pointer is equal to or greater than 3 (4 bits received). For RFC and ARG frames, the signal is true after the input pointer reaches 5 (6 bits received).

TABLE 4.1: CHIP SIGNAL DESCRIPTION (continued)

IRC1-3	Input register control bits C1-3. These are the three control bits of the received frame after it has been loaded into the input register.
LDR2	Load register 2. This signal strobes the received frame from the input register into register 2 and the upper three bits of register 1.
LR2P2	Load register 2 at phase 2. This is just the LDR2 signal clocked by the phase 2 clock for PLA synchronization.
OPT11P2	Output pointer at 11, clocked by phase 2. This signal indicates that the output pointer has cycled through all 11 bits. The output pointer controls both frame transmission and error-checking. Phase 2 simply gates the signal for input to the PLA.
READ	Read indicates that one of the registers is being read out to the microprocessor bus. This signal inhibits LDR2 so that data does not change midway through a read operation.
RLIR	Receiver load input register. This signal gates the incoming frame from the input buffer into the input register. RLIR goes true when a frame is coming in and the acceptor is in the idle state (AIDS) and the driver is not in the auto retransmit state (DTRS). This situation indicates the input register is available since the incoming frame has either already been loaded into R1R and R2R or completely retransmitted. So RLIR=AIDS.$\overline{\text{DTRS}}$.
RTAS	Receiver transfer abort state. RTAS indicates that a second sync bit has been received in the middle of the incoming frame.
STRSP2	Source transfer state, clocked by phase 2. This signal indicates that the output register has been written to.

The chip acceptor begins operation when RLIR goes true in response to the incoming frame. From ALIR, several paths are possible depending on the frame type and the chip status.

If the receiver detects a second sync pulse in the middle of a frame (RTAS=1), the state machine immediately aborts back to AIDS.

If the decode logic determines that the incoming frame must be automatically retransmitted, the acceptor transitions to the

ATRS state. The sync bit causes this transition for data frames and the IPGE35 signal causes it for the other frames which need to be retransmitted.

If the chip is in analyzer mode (TA=LA=1), ALIR goes to AIFS immediately.

These first three paths out of ALIR occur as soon as the condition is true and they do not wait for the entire frame to be received. The remaining four transitions out of ALIR do not happen until the whole frame has been received into the input register. This is indicated by RLIR going low.

If the frame is not automatically retransmitted and requires CPU interpretation, ALIR goes to ADYS. This state sets TORAV, which in turn sets ORAV when the acceptor returns to AIDS. If register 2 is empty (FRAV+FRNS=FRMR2=0), then ADYS goes to ACDS. If FRMR2 is still high, this transition is held off until the CPU reads register 2 (resetting FRAV or FRNS). ACDS causes LDR2 to go true, which loads the frame from the input register into R2R and the upper three bits of R1R, and also sets FRAV or FRNS (but not both), depending on the state of the EQ signal. Note that ACDS can be reached from ATRS also. If the frame is in the CMDI group, it must be retransmitted (ATRS) and loaded into R1 and R2 (ACDS) for the CPU.

When the received frame requires error-checking, ALIR goes to AECS. EQ is set true in ALIR and will remain so unless error-checking detects an error in AECS. EQ determines whether the LDR2 signal sets FRAV (EQ still high) or FRNS (EQ set low indicating an error) when the acceptor reaches ACDS. Note that the only path from AECS to ACDS requires EQ to be low (error). This path goes through AHSS (which initiates the sequence to set ORAV) and ADYS to ACDS, which loads the bad frame into R1R and R2R and sets FRNS (indirectly). All other paths out of AECS have EQ high (no error). If a non-CMD frame error-checks OK, the transition is to AHSS (sets ORAV indirectly) and back to AIDS. If ■ CMD frame error-checks correctly, the transition is to ARFC. This state links to the driver state machine to cause the transmission of an RFC frame. This is the auto RFC feature for the loop controller. If the output register is written while in AECS, error-checking is aborted and the state machine returns to AIDS.

The transition from ALIR to AHSS occurs under two conditions. If the device is controller active and an RFC frame is received, or if the device is controller active but is not the system controller and an IFC is received (IFCR set), AHSS sets ORAV (via TORAV) and the acceptor returns to the idle state. In the first case, ORAV indicates that the CMD-RFC handshake is complete and the controller may source its next frame. In the second case, IFCR signals the controller to reset itself (especially CA=0) and then to retransmit the IFC by writing to the output register.

When an RFC is received in a non-controller device and the LRDY signal is not already true, ALIR goes to AWRS. When the device writes SLRDY, which sets LRDY, the acceptor goes to ARFC

which causes the driver to send (actually, retransmit) the RFC.

In general, acceptor operation moves from AIDS to ALIR and through the various paths back to AIDS without abnormal delays. There are two situations, however, where the chip operation will hang up waiting for CPU action. If the acceptor is in AWRS, it will remain there until the CPU sets LRDY by writing the SLRDY bit high. Also, if the acceptor is in ADYS, but the CPU has not read the previous frame from R2R, no transition will occur until the read is performed. If another frame is received while the acceptor is waiting in one of these two states, it will remain in the input buffer (RLIR cannot go true since the state AIDS is false) unless it is an IDY or CMD (IBIC signal true). In this case (IDY or CMD), it is desirable to abort AWRS or ADYS and return to AIDS so the incoming frame can be processed. When this occurs, the frame previously in the input register is lost. This is the only situation where the frame in the input register can be destroyed. If the incoming frame is other than IDY or CMD, the input register is preserved, and the only the input buffer is overwritten by the most recent frame. The acceptor remains in a hung state until the CPU takes appropriate action.

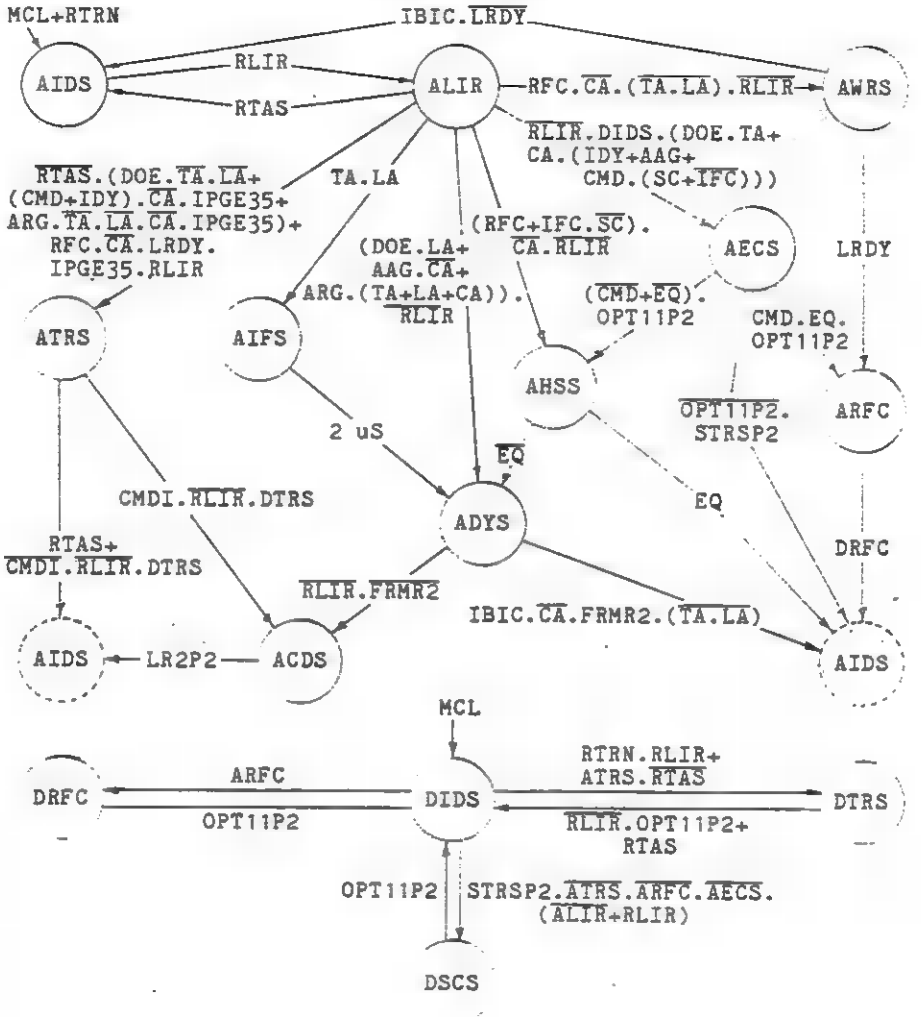
The following pages contain the state diagrams for all the important state machines in the chip. The acceptor state diagram has been described in detail in the preceeding text. The other state diagrams are relatively simple and require no additional explanation.

There are basically four HP-IL interface functions which handle the reception and transmission of frames: the Receiver interface function, the Acceptor Handshake interface function, the Source Handshake interface function, and the Driver interface function. It is important to remember that these interface functions do not have direct counterparts in the chip state machines.

The chip acceptor state machine, for example, in combination with the CPU performs the operations of both the Receiver and Acceptor Handshake interface functions as well as part of the Source Handshake interface function.

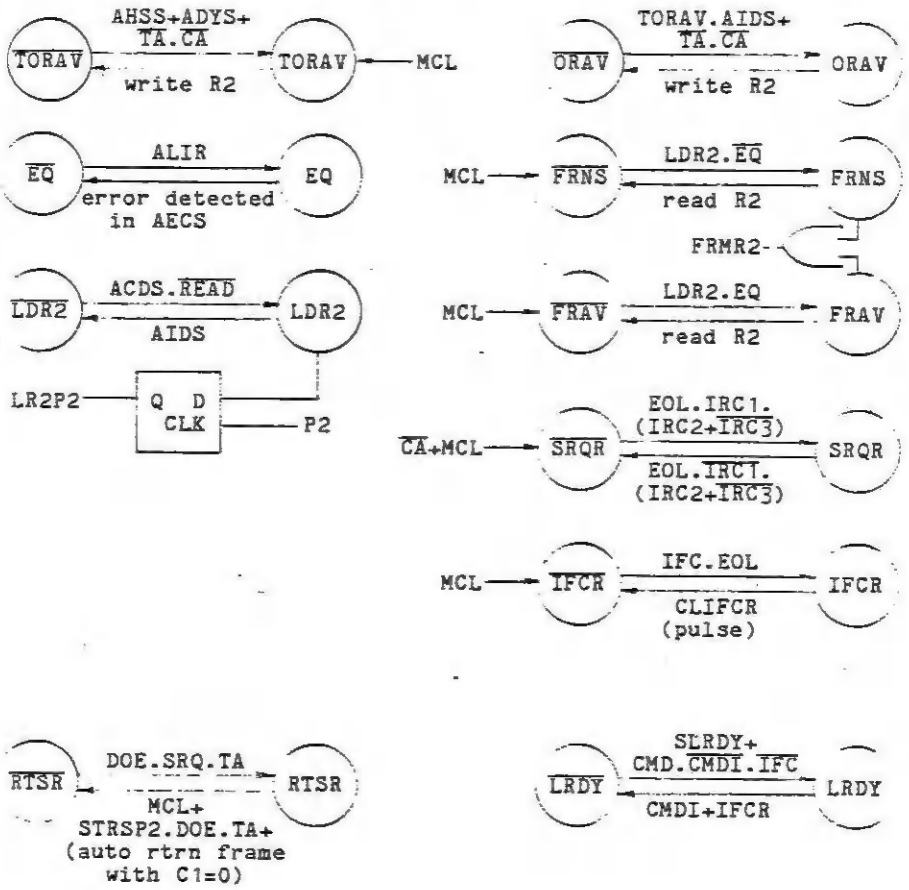
While the chip driver state machine plus CPU corresponds fairly closely in function to the Driver interface function, the states within the state machine do not all have the same function as the states within the Driver interface function. DTRS is very much the same as the interface function state {DTRS}. This state handles automatic retransmission of received frames. The interface function state {DSCS} is identical in name to the chip state DSCS, but performs a different function. {DSCS} is only used to source frames, for example, a talker sending DAB frames. {DACS} is used to retransmit frames which have been received and processed by this device. On the chip, however, DSCS is used for any frame written to the output register, which includes both sourcing and retransmitting processed frames. DRFC is used both to source the RFC frame (done by {DSCS}) and to retransmit the RFC frame (performed by {DACS}).

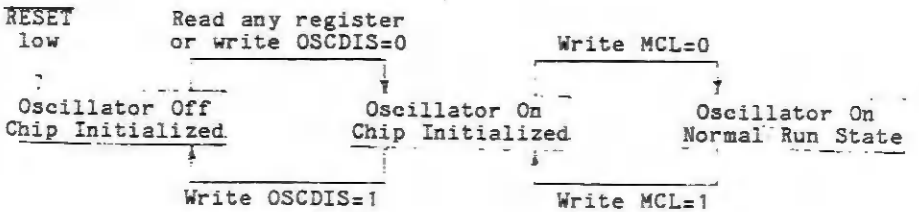
FIGURE 4.2: CHIP ACCEPTOR AND DRIVER STATE DIAGRAMS



AIDS - acceptor idle state
ALIR - acceptor load input register state
ATRS - acceptor transfer state
ADYS - acceptor delay state
ACDS - acceptor data state
AECS - acceptor error-check state
AHSS - acceptor handshake state
AIFS - acceptor input frame state
AWRS - acceptor wait for local ready state
ARFC - acceptor RFC state
DIDS - driver idle state
DRFC - driver RFC state
DSCS - driver source state
DTRS - driver transfer state

FIGURE 4.3: CHIP INTERRUPT FLAG AND MISCELLANEOUS STATE DIAGRAMS





MCL sets IFCR=SRQR=FRNS=FRAV=0, ORAV=1, interrupt enable bits=0, SC=SCTL, RTSR latch clear, ORE=1.

ACKNOWLEDGMENT

Initial work on the definition of HP-IL and the interface chip was done by Tom Heger and Dave Sweetser. The actual logic design and circuit design of the chip was done by Mike Pan. My own contribution was small, involving some minor design changes to permit the chip to be used in a 28 pin package and a general strengthening of the peripheral circuitry for greater protection from electrostatic discharge and latch-up.

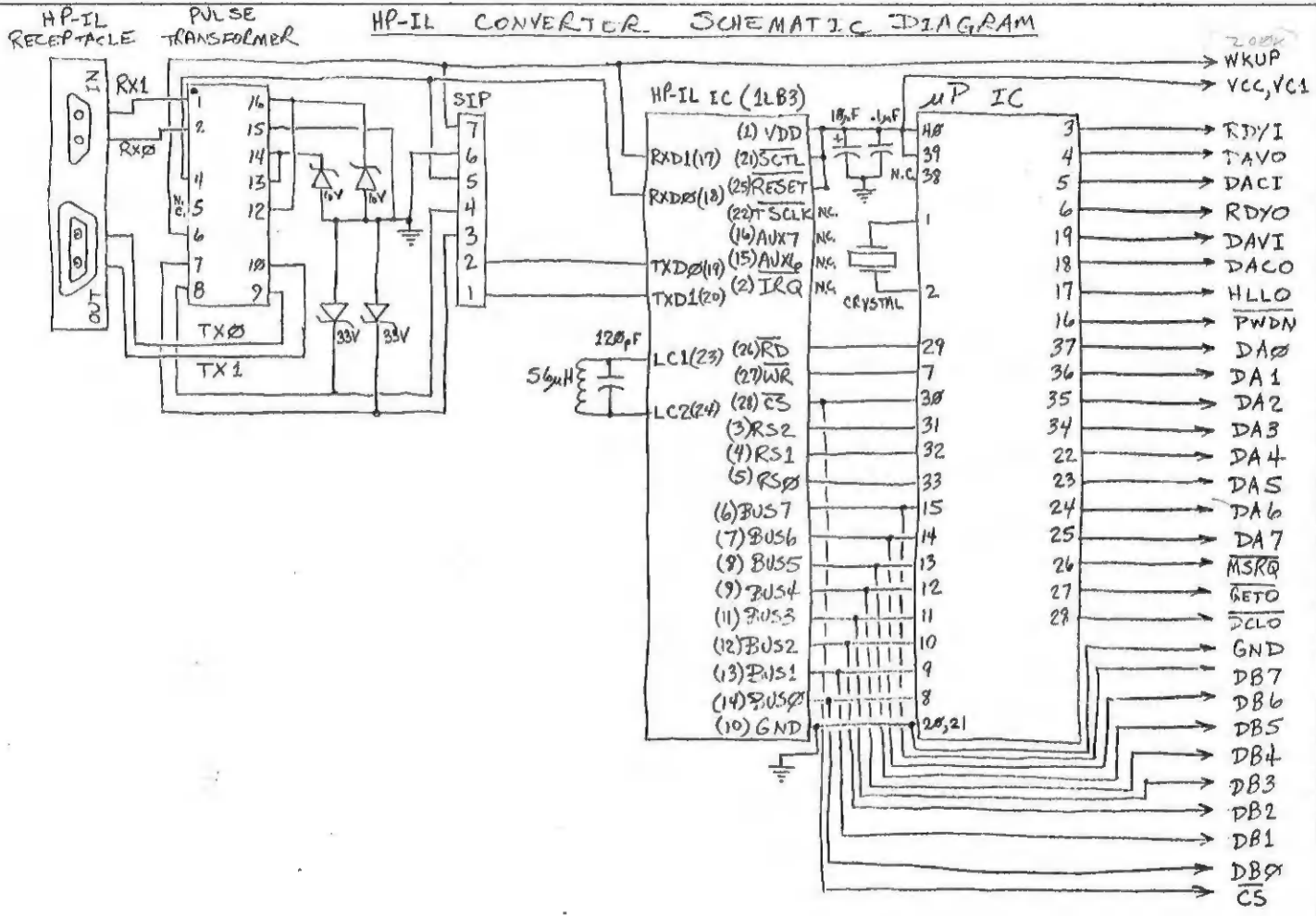
Steve Harper
Corvallis Division
Hewlett-Packard Co.
1981 MAY 15

- One of the chip pins enables a repeater mode where the device is electrically (but not physically) removed from the loop (40 pin version only).

The HP-IL driver and receiver circuitry on the chip is isolated from the actual interface lines with pulse transformers and a few discrete components. The chip has an oscillator which requires an external LC network for frequency control. The connection to the CPU bus and associated control lines is usually direct.

The CPU communicates with the chip through simple memory or I/O read and write cycles. The chip has a chip select line and three address lines to permit data transfer to or from eight locations or registers on the chip. In addition to the data transfer, reading or writing certain of the registers causes other actions to take place, such as the transmission of a frame over the interface loop. When CPU action is necessary, the chip has an interrupt line to notify the microprocessor.

The 28 pin plastic DIP version (1LB3-0003) is presently available from Corvallis Division. The 40 pin ceramic DIP version is not presently qualified, but could be qualified and made available if there is sufficient need. It is felt in most cases that the additional pins in the 40 pin version will not be needed. The expense of the 40 pin package and the expense of qualification is strong reason to choose the 28 pin package.


HP-IL RECEPTACLE
PULSE TRANSFORMER
HP-IL CONVERTER SCHEMATIC DIAGRAM
IN
OUT
RX1
RXØ
TXØ
TX1
16V
16V
33V
33V
SIP
HP-IL IC (1LB3)
μP IC
(1) VDD
RXD1(17)
(21) SCTL
(25) RESET
RXDØ(18)
(22) SCLK
(16) AUX7
TXDØ(19)
(15) AUXØ
TXD1(20)
(2) IRQ
CRYSTAL
N.C.
18pF
.1μF
120pF
56μH
LC1(23)
LC2(24)
(26) RD
(27) WR
(28) CS
(3) RS2
(4) RS1
(5) RSØ
(6) BUS7
(7) BUS6
(8) BUS5
(9) BUS4
(11) BUS3
(12) BUS2
(13) BUS1
(14) BUSØ
(10) GND
HØ
WKUP
VCC,VC1
RDYI
TAVO
DACI
RDYO
DAVI
DACO
HLLO
PWDN
DAØ
DA1
DA2
DA3
DA4
DA5
DA6
DA7
MSRQ
GETO
DCLO
GND
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DBØ
CS